

BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL

```
BBBBBBBBB      AAAAAA      SSSSSSSSS      SSSSSSSSS      LL      EEEEEEEEEEE      EEEEEEEEEEE      PPPPPPPP
BBBBBBBBB      AAAAAA      SSSSSSSSS      SSSSSSSSS      LL      EEEEEEEEEEE      EEEEEEEEEEE      PPPPPPPP
BB          BB  AA          AA  SS          SS  LL          LL          PP          PP
BB          BB  AA          AA  SS          SS  LL          LL          PP          PP
BB          BB  AA          AA  SS          SS  LL          LL          PP          PP
BBBBBBBBB      AA          AA  SSSSSSS      SSSSSSS      LL          LL          PPPPPPPP
BBBBBBBBB      AA          AA  SSSSSSS      SSSSSSS      LL          LL          PPPPPPPP
BB          BB  AAAAAAAAAA      SS          SS  LL          LL          PP
BB          BB  AAAAAAAAAA      SS          SS  LL          LL          PP
BB          BB  AA          AA  SSSSSSS      SSSSSSS      LL          LL          PP
BB          BB  AA          AA  SSSSSSS      SSSSSSS      LL          LL          PP
BBBBBBBBB      AA          AA  SSSSSSS      SSSSSSS      LLLLLLLLLLL      EEEEEEEEEEE      EEEEEEEEEEE      PP
BBBBBBBBB      AA          AA  SSSSSSS      SSSSSSS      LLLLLLLLLLL      EEEEEEEEEEE      EEEEEEEEEEE      PP
                                                                ....
                                                                ....
                                                                ....
                                                                ....

LL          IIIIIII      SSSSSSSSS
LL          IIIIIII      SSSSSSSSS
LL          II          SS
LL          II          SS
LL          II          SS
LL          II          SS
LL          II          SSSSSSS
LL          II          SSSSSSS
LL          II          SS
LL          II          SS
LL          II          SS
LL          II          SS
LLLLLLLLLLL      IIIIIII      SSSSSSSSS
LLLLLLLLLLL      IIIIIII      SSSSSSSSS
```

```

1 0001 0 MODULE BASSSLEEP (
2 0002 0 IDENT = '3-003'
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: VAX-11 BASIC Miscellaneous I/O
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module contains the BASIC SLEEP function.
36 0036 1
37 0037 1 ENVIRONMENT: VAX-11 User Mode
38 0038 1
39 0039 1 AUTHOR: John Sauter, CREATION DATE: 19-APR-1979, REWRITTEN: 11-JUN-1980
40 0040 1 REWRITTEN by Farokh Morshed 18-NOV-81.
41 0041 1
42 0042 1 MODIFIED BY:
43 0043 1
44 0044 1 2-001 - Rewrite this routine to use $QIO instead of RMS' read-with
45 0045 1 -timeout. The previous version was 1-005. JBS 11-JUN-1980
46 0046 1 2-002 - Designate this version 2-002 to keep version numbers consistent
47 0047 1 since it is the "enhancement" version of BASSSLEEP. JBS 12-JUN-1980
48 0048 1 3-001 - To implement type-ahead recovery, and SYSSINPUT translation using
49 0049 1 SPARSE, This module was rewritten. Farokh Morshed 18-NOV-81.
50 0050 1 3-002 - Get rid of all $TRNLOG code since system services do that now.
51 0051 1 FM 14-DEC-81.
52 0052 1 3-003 - Put in the code to get event flags from LIB$GET_EF so event flag
53 0053 1 zero is left alone. FM 29-JUN-82.
54 0054 1 --
55 0055 1
56 0056 1 !<BLF/PAGE>

```

```

! Sleep for a while
! File: BASSSLEEP.B32, Edit: FM3003

```

```
58 0057 1 |
59 0058 1 | SWITCHES:
60 0059 1 |
61 0060 1 |
62 0061 1 | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
63 0062 1 |
64 0063 1 |
65 0064 1 | LINKAGES:
66 0065 1 |
67 0066 1 |     NONE
68 0067 1 |
69 0068 1 | TABLE OF CONTENTS:
70 0069 1 |
71 0070 1 |
72 0071 1 | FORWARD ROUTINE
73 0072 1 |     BASSSLEEP : NOVALUE,
74 0073 1 |     TAKE_AST : NOVALUE,
75 0074 1 |     SLEEP_HANDLER;
76 0075 1 |
77 0076 1 |
78 0077 1 | INCLUDE FILES:
79 0078 1 |
80 0079 1 |
81 0080 1 | REQUIRE 'RTLIN:RTLPSECT';
82 0175 1 |
83 0176 1 | LIBRARY 'RTLSTARLE';
84 0177 1 |
85 0178 1 |
86 0179 1 | MACROS:
87 0180 1 |
88 0181 1 |     NONE
89 0182 1 |
90 0183 1 | EQUATED SYMBOLS:
91 0184 1 |
92 0185 1 |     NONE
93 0186 1 |
94 0187 1 | PSECTS:
95 0188 1 |
96 0189 1 | DECLARE_PSECTS (BAS);
97 0190 1 |
98 0191 1 | OWN STORAGE:
99 0192 1 |
100 0193 1 |     NONE
101 0194 1 |
102 0195 1 | EXTERNAL REFERENCES:
103 0196 1 |
104 0197 1 |
105 0198 1 | EXTERNAL ROUTINE
106 0199 1 |     LIB$MATCH_COND,
107 0200 1 |     LIB$GET_EF,
108 0201 1 |     LIB$FREE_EF,
109 0202 1 |     LIB$STOP;
110 0203 1 |
```

! Wait some seconds
! Take the AST for QIO or SETIMR
! Clean up after UNWINDs

! Macros for defining psects
! System symbols

! Declare psects for BASS\$ facility

! Match condition codes
! Get event flag
! Free event flag
! Signal a fatal error

```
112 0204 1 GLOBAL ROUTINE BASSSLEEP (          ! Wait a while
113 0205 1     SECONDS                        ! Seconds to wait
114 0206 1     ) : NOVALUE =
115 0207 1
116 0208 1 ++
117 0209 1 FUNCTIONAL DESCRIPTION:
118 0210 1
119 0211 1     Wait the specified number of seconds, or until a terminator
120 0212 1     is typed on the controlling terminal.
121 0213 1
122 0214 1 FORMAL PARAMETERS:
123 0215 1
124 0216 1     SECONDS.rl.v     How many seconds to wait.
125 0217 1
126 0218 1 IMPLICIT INPUTS:
127 0219 1
128 0220 1     NONE
129 0221 1
130 0222 1 IMPLICIT OUTPUTS:
131 0223 1
132 0224 1     NONE
133 0225 1
134 0226 1 ROUTINE VALUE:
135 0227 1 COMPLETION CODES:
136 0228 1
137 0229 1     NONE
138 0230 1
139 0231 1 SIDE EFFECTS:
140 0232 1
141 0233 1     Signals if an error is encountered.
142 0234 1
143 0235 1 --
144 0236 1
145 0237 2 BEGIN
146 0238 2
147 0239 2 LOCAL
148 0240 2     ASSIGNED_CHAN : VOLATILE,          !Assigned channel. Zero if no channel is assigned.
149 0241 2     TIMER_REQID : VOLATILE,           !$SETIMR request ID.
150 0242 2     EVENT_FLAG : VOLATILE,           !Event flag to be used instead of Ef zero.
151 0243 2     EF_STATUS;
152 0244 2
153 0245 2 ++
154 0246 2 Arrange to clean up if an UNWIND is done. The most likely cause of an
155 0247 2 UNWIND is a ^C from the $HIBER.
156 0248 2 --
157 0249 2
158 0250 2 ENABLE
159 0251 2     SLEEP_HANDLER (ASSIGNED_CHAN, TIMER_REQID, EVENT_FLAG);
160 0252 2
161 0253 2 ++
162 0254 2 Get an event flag to be used from here on. We do this so event flag zero
163 0255 2 can be used by others. We never use this event flag for anything.
164 0256 2 --
165 0257 2     EF_STATUS = LIB$GET_EF (EVENT_FLAG);
166 0258 2     IF (NOT .EF_STATUS) THEN LIB$STOP(.EF_STATUS);
167 0259 2
168 0260 2 !+
```

```
169 0261 2 ! Make sure there is not a $WAKE hanging around.
170 0262 2 !-
171 0263 2 BEGIN
172 0264 2 LOCAL
173 0265 2 WAKE_STATUS,
174 0266 2 HIBER_STATUS;
175 0267 2
176 0268 2 WAKE_STATUS = $WAKE ();
177 0269 2
178 0270 2 IF ( NOT .WAKE_STATUS) THEN LIB$STOP (.WAKE_STATUS);
179 0271 2
180 0272 2 HIBER_STATUS = $HIBER;
181 0273 2
182 0274 2 IF ( NOT .HIBER_STATUS) THEN LIB$STOP (.HIBER_STATUS);
183 0275 2
184 0276 2 END;
185 0277 2 BEGIN
186 0278 2 BUILTIN
187 0279 2 EMUL;
188 0280 2
189 0281 2 LOCAL
190 0282 2 SETIMR_STATUS,
191 0283 2 TIMBUF : VECTOR [2];
192 0284 2
193 0285 2 !Translated seconds for $SETIMR.
194 0286 2
195 0287 2 !+
196 0288 2 Compute time to wake in system format
197 0289 2 !-
198 0290 2 EMUL (%REF (-10000000), SECONDS, %REF (0), TIMBUF [0]);
199 0291 2 !+
200 0292 2 Take an AST when that time comes.
201 0293 2 We will pick address of SECONDS to be our TIMER_REQID since this address
202 0294 2 is unique for each call.
203 0295 2 !-
204 0296 2 TIMER_REQID = SECONDS;
205 0297 2 SETIMR_STATUS = $SETIMR (EFN = .EVENT_FLAG, DAYTIM = TIMBUF [0], ASTADR = TAKE_AST, REQIDT = TIMER_REQID
206 0298 2
207 0299 2 IF ( NOT .SETIMR_STATUS) THEN LIB$STOP (.SETIMR_STATUS);
208 0300 2
209 0301 2 END;
210 0302 2 !+
211 0303 2 Stop early if a line terminator is typed.
212 0304 2 !-
213 0305 2 BEGIN
214 0306 2 LOCAL
215 0307 2 DEVCHR : BLOCK [DIB$K_LENGTH, BYTE],
216 0308 2 DEVCHR_DESC : BLOCK [8, BYTE],
217 0309 2 GETDEV_STATUS,
218 0310 2 TRNNAM_DESC : BLOCK [8, BYTE];
219 0311 2
220 0312 2 TRNNAM_DESC [DSC$W_LENGTH] = %CHARCOUNT('SYSS$INPUT');
221 0313 2 TRNNAM_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_2;
222 0314 2 TRNNAM_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
223 0315 2 TRNNAM_DESC [DSC$A_POINTER] = UPLIT('SYSS$INPUT');
224 0316 2 !+
225 0317 2
```

```
226 0318 3 ! Do a $GETDEV on this device name.
227 0319 3 !-
228 0320 3     DEVCHR_DESC [DSC$W_LENGTH] = DIB$K_LENGTH;
229 0321 3     DEVCHR_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_Z;
230 0322 3     DEVCHR_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
231 0323 3     DEVCHR_DESC [DSC$A_POINTER] = DEVCHR;
232 0324 3     GETDEV_STATUS = $GETDEV (DEVNAM = TRNNAM_DESC, PRILEN = DEVCHR_DESC [DSC$W_LENGTH], PRIBUF = DEVCHR_DESC
233 0325 3     ;
234 0326 3
235 0327 3     IF (.DEVCHR [DIB$B_DEVCLASS] EQL DC$_TERM)
236 0328 3     THEN
237 0329 3 +
238 0330 3     SYSS$INPUT is a terminal. Arrange to take an AST if a terminator is typed
239 0331 3     on it.
240 0332 3     !-
241 0333 3     BEGIN
242 0334 3
243 0335 3     LOCAL
244 0336 3     QIO_STATUS,
245 0337 3     ASSIGN_STATUS;
246 0338 3
247 0339 3     ASSIGN_STATUS = $ASSIGN (DEVNAM = TRNNAM_DESC, CHAN = ASSIGNED_CHAN);
248 0340 3
249 0341 3     IF ( NOT .ASSIGN_STATUS) THEN LIB$STOP (.ASSIGN_STATUS);
250 0342 3
251 P 0343 3     QIO_STATUS = $QIO (EFN = .EVENT_FLAG, CHAN = .ASSIGNED_CHAN, FUNC = (IOS$_SETMODE OR IOS$_OUTBAND OR
252 0344 3     P1 = TAKE_AST, P2 = UPLIT (0, %X'2000')); !Terminator is a CR.
253 0345 3
254 0346 3     IF ( NOT .QIO_STATUS) THEN LIB$STOP (.QIO_STATUS);
255 0347 3
256 0348 3     END;
257 0349 3
258 0350 3     END;
259 0351 3 +
260 0352 3     Now wait for the $SETIMR to fire, or (if SYSS$INPUT is a terminal)
261 0353 3     for a terminator to be typed.
262 0354 3     !-
263 0355 3     BEGIN
264 0356 3
265 0357 3     LOCAL
266 0358 3     HIBER_STATUS;
267 0359 3
268 0360 3     HIBER_STATUS = $HIBER;
269 0361 3
270 0362 3     IF ( NOT .HIBER_STATUS) THEN LIB$STOP (.HIBER_STATUS);
271 0363 3
272 0364 3     END;
273 0365 3 +
274 0366 3     At this point either AST for $SETIMR or $QIO has gone off. We don't care
275 0367 3     which, we just cancel both of them, and also deassign the channel.
276 0368 3     !-
277 0369 3     BEGIN
278 0370 3
279 0371 3     LOCAL
280 0372 3     DASSGN_STATUS,
281 0373 3     CANTIM_STATUS,
282 0374 3     QIO_STATUS;
```

BASSSLEEP
3-003

N 15
16-Sep-1984 01:14:56
14-Sep-1984 11:56:40

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASSSLEEP.B32;1

Page 6
(3)

```

: 283      0375      3
: 284      0376      3
: 285      0377      3
: 286      0378      3
: 287      0379      3
: 288      0380      3
: 289      0381      3
: 290      0382      4
: 291      P 0383      4
: 292      0384      4
: 293      0385      4
: 294      0386      4
: 295      0387      4
: 296      0388      4
: 297      0389      4
: 298      0390      4
: 299      0391      4
: 300      0392      3
: 301      0393      3
: 302      0394      3
: 303      0395      3
: 304      0396      3
: 305      0397      3
: 306      0398      3
: 307      0399      3
: 308      0400      3
: 309      0401      3
: 310      0402      3
: 311      0403      3
: 312      0404      3
: 313      0405      3
: 314      0406      3
: 315      0407      3
: 316      0408      3
: 317      0409      3
: 318      0410      3
: 319      0411      3
: 320      0412      3
: 321      0413      3
: 322      0414      3
: 323      0415      3
: 324      0416      3
: 325      0417      3
: 326      0418      2
: 327      0419      2
: 328      0420      2
: 329      0421      2
: 330      0422      1

CANTIM_STATUS = $CANTIM (REQIDT = TIMER_REQID);
IF ( NOT .CANTIM_STATUS) THEN LIB$STOP (.CANTIM_STATUS);
IF .ASSIGNED_CHAN NEQ 0
THEN
  BEGIN
    QIO_STATUS = $QIO (EFN = .EVENT_FLAG, CHAN = .ASSIGNED_CHAN, FUNC = (IO$_SETMODE OR IO$_OUTBAND OR
      P1 = 0));
    IF ( NOT .QIO_STATUS) THEN LIB$STOP (.QIO_STATUS);
    DASSGN_STATUS = $DASSGN (CHAN = .ASSIGNED_CHAN);
    IF ( NOT .DASSGN_STATUS) THEN LIB$STOP (.DASSGN_STATUS);
  END;
END;
+ Make sure there are not any $WAKE hanging around. They could have appeared
+ as a result of one of the ASTs timer or QIO going off just before we turned
- it off.
  BEGIN
    LOCAL
      WAKE_STATUS,
      HIBER_STATUS;
    WAKE_STATUS = $WAKE ();
    IF ( NOT .WAKE_STATUS) THEN LIB$STOP (.WAKE_STATUS);
    HIBER_STATUS = $HIBER;
    IF ( NOT .HIBER_STATUS) THEN LIB$STOP (.HIBER_STATUS);
  END;
+ Free the event flag now
- EF_STATUS = LIB$FREE_EF (EVENT_FLAG);
  IF (NOT .EF_STATUS) THEN LIB$STOP (.EF_STATUS);
RETURN;
END;
```

! end of BASSSLEEP

.TITLE BASSSLEEP
.IDENT \3-003\

.PSECT _BASSCODE,NOWRT, SHR, PIC,2

00 00 00 54 55 50 4E 49 24 53 59 53 00000 P.AAA: .ASCII \SYSS\$INPUT\<0><0><0>
00002000 00000000 0000C P.AAB: .LONG 0, 8192

MOVAB	SYSSQIO, R6	
MOVAB	SYSSWAKE, R5	
MOVAB	SYSSHIBER, R4	
MOVAB	LIB\$STOP, R3	
MOVAB	-144(SP), SP	
CLRQ	EVENT FLAG	0237
CLRL	ASSIGNED CHAN	
MOVAL	14\$, (FP)	
PUSHAB	EVENT FLAG	0257
CALLS	#1, LIB\$GET_EF	
MOVL	R0, EF_STATUS	
BLBS	EF_STATUS, 1\$	0258
PUSHL	EF_STATUS	
CALLS	#1, LIB\$STOP	
CLRQ	-(SP)	0269
CALLS	#2, SYSSWAKE	
BLBS	WAKE_STATUS, 2\$	0271
PUSHL	WAKE_STATUS	
CALLS	#1, LIB\$STOP	
CALLS	#0, SYSSHIBER	0273
BLBS	HIBER_STATUS, 3\$	0275
PUSHL	HIBER_STATUS	
CALLS	#1, LIB\$STOP	
EMUL	#-10000000, SECONDS, #0, TIMBUF	0290
MOVAB	SECONDS, TIMER_REQID	0296
PUSHAB	TIMER_REQID	0297
PUSHAB	TAKE_AST	
PUSHAB	TIMBUF	
PUSHL	EVENT FLAG	
CALLS	#4, SYSSSETIMR	
BLBS	SETIMR_STATUS, 4\$	0299
PUSHL	SETIMR_STATUS	
CALLS	#1, LIB\$STOP	
MOVL	#16777225, TRNNAM_DESC	0313
MOVAB	P.AAA, TRNNAM_DESC+4	0316
MOVL	#16777332, DEVCHR_DESC	0320
MOVAB	DEVCHR, DEVCHR_DESC+4	0323
CLRQ	-(SP)	0324
PUSHAB	DEVCHR_DESC	
PUSHAB	DEVCHR_DESC	
PUSHAB	TRNNAM_DESC	
CALLS	#5, SYSSGETDEV	
CMPB	DEVCHR+4, #66	0327
BNEQ	6\$	
CLRQ	-(SP)	0339
PUSHAB	ASSIGNED CHAN	
PUSHAB	TRNNAM_DESC	
CALLS	#4, SYSSASSIGN	

05	50	E8	000C9	BLBS	ASSIGN_STATUS, 5\$: 0341
	50	DD	000CC	PUSHL	ASSIGN_STATUS	:
63	01	FB	000CE	CALLS	#1, LIB\$STOP	: 0344
	7E	7C	000D1	5\$: CLRQ	-(SP)	:
	7E	7C	000D3	CLRQ	-(SP)	:
	FF1F	CF	9F	PUSHAB	P.AAB	:
	0000V	CF	9F	PUSHAB	TAKE_AST	:
		7E	7C	CLRQ	-(SP)	:
		7E	D4	CLRL	-(SP)	:
7E	0C23	8F	3C	MOVZWL	#3107, -(SP)	:
	FC	AD	DD	PUSHL	ASSIGNED_CHAN	:
	F4	AD	DD	PUSHL	EVENT_FLAG	:
66	0C	FB	000EC	CALLS	#12, SYSSQIO	:
05	50	E8	000EF	BLBS	QIO_STATUS, 6\$: 0346
	50	DD	000F2	PUSHL	QIO_STATUS	:
63	01	FB	000F4	CALLS	#1, LIB\$STOP	:
64	00	FB	000F7	6\$: CALLS	#0, SYSSHIBER	: 0360
05	50	E8	000FA	BLBS	HIBER_STATUS, 7\$: 0362
	50	DD	000FD	PUSHL	HIBER_STATUS	:
63	01	FB	000FF	CALLS	#1, LIB\$STOP	:
	7E	D4	00102	7\$: CLRL	-(SP)	: 0376
	F8	AD	9F	PUSHAB	TIMER_REQID	:
00000000G	00	02	FB	CALLS	#2, SYSSCANTIM	:
05	50	E8	0010E	BLBS	CANTIM_STATUS, 8\$: 0378
	50	DD	00111	PUSHL	CANTIM_STATUS	:
63	01	FB	00113	CALLS	#1, LIB\$STOP	:
	FC	AD	D5	8\$: TSTL	ASSIGNED_CHAN	: 0380
		32	13	BEQL	10\$:
		7E	7C	CLRQ	-(SP)	: 0384
		7E	7C	CLRQ	-(SP)	:
		7E	7C	CLRQ	-(SP)	:
		7E	7C	CLRQ	-(SP)	:
		7E	D4	CLRL	-(SP)	:
7E	0C23	8F	3C	MOVZWL	#3107, -(SP)	:
	FC	AD	DD	PUSHL	ASSIGNED_CHAN	:
	F4	AD	DD	PUSHL	EVENT_FLAG	:
66	0C	FB	00130	CALLS	#12, SYSSQIO	:
05	50	E8	00133	BLBS	QIO_STATUS, 9\$: 0386
	50	DD	00136	PUSHL	QIO_STATUS	:
63	01	FB	00138	CALLS	#1, LIB\$STOP	:
	FC	AD	DD	9\$: PUSHL	ASSIGNED_CHAN	: 0388
00000000G	00	01	FB	CALLS	#1, SYSSDASSGN	:
05	50	E8	00145	BLBS	DASSGN_STATUS, 10\$: 0390
	50	DD	00148	PUSHL	DASSGN_STATUS	:
63	01	FB	0014A	CALLS	#1, LIB\$STOP	:
	7E	7C	0014D	10\$: CLRQ	-(SP)	: 0406
65	02	FB	0014F	CALLS	#2, SYSSWAKE	:
05	50	E8	00152	BLBS	WAKE_STATUS, 11\$: 0408
	50	DD	00155	PUSHL	WAKE_STATUS	:
63	01	FB	00157	CALLS	#1, LIB\$STOP	:
64	00	FB	0015A	11\$: CALLS	#0, SYSSHIBER	: 0410
05	50	E8	0015D	BLBS	HIBER_STATUS, 12\$: 0412
	50	DD	00160	PUSHL	HIBER_STATUS	:
63	01	FB	00162	CALLS	#1, LIB\$STOP	:
	F4	AD	9F	12\$: PUSHAB	EVENT_FLAG	: 0418
00000000G	00	01	FB	CALLS	#1, LIB\$FREE_EF	:
52	50	D0	0016F	MOVL	R0, EF_STATUS	:

D 16
16-Sep-1984 01:14:56 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:56:40 [BASRTL.SRC]BASSLEEP.B32;1

Page 9
(3)

05		52	E8	00172	BLBS	EF_STATUS, 13\$
		52	DD	00175	PUSHL	EF_STATUS
63		01	FB	00177	CALLS	#1, LIB\$STOP
			04	0017A	RET	
		0000		0017B	.WORD	Save nothing
50	08	AC	DD	0017D	MOVL	8(AP), R0
50	04	A0	DD	00181	MOVL	4(R0), R0
	F4	A0	9F	00185	PUSHAB	EVENT_FLAG
	F8	A0	9F	00188	PUSHAB	TIMER_REQID
	FC	A0	9F	0018B	PUSHAB	ASSIGNED_CHAN
		03	DD	0018E	PUSHL	#3
		5E	DD	00190	PUSHL	SP
7E	04	AC	7D	00192	MOVQ	4(AP), -(SP)
0000V	CF	03	FB	00196	CALLS	#3, SLEEP_HANDLER
			04	0019B	RET	

0419
0422
0237

```
; Routine Size: 412 bytes,    Routine Base: _BAS$CODE + 0014
```

: 331 0423 1

```
: 333      0424 1 ROUTINE TAKE_AST (           ! Take a QIO, or SETIMR AST.
: 334      0425 1     AST_PARAM           ! AST parameter.
: 335      0426 1     ) : NOVALUE =
: 336      0427 1
: 337      0428 1 ++
: 338      0429 1 FUNCTIONAL DESCRIPTION:
: 339      0430 1
: 340      0431 1     Take an AST, either from $SETIMR when the sleep time is up, or from
: 341      0432 1     the $QIO when it completes. In both cases we simply do a $WAKE.
: 342      0433 1
: 343      0434 1 FORMAL PARAMETERS:
: 344      0435 1
: 345      0436 1     AST_PARAM           Pointer to parameters for this AST.
: 346      0437 1
: 347      0438 1 IMPLICIT INPUTS:
: 348      0439 1
: 349      0440 1     NONE
: 350      0441 1
: 351      0442 1 IMPLICIT OUTPUTS:
: 352      0443 1
: 353      0444 1     NONE
: 354      0445 1
: 355      0446 1 COMPLETION CODES:
: 356      0447 1
: 357      0448 1     NONE
: 358      0449 1
: 359      0450 1 SIDE EFFECTS:
: 360      0451 1
: 361      0452 1     NONE
: 362      0453 1
: 363      0454 1 --
: 364      0455 1
: 365      0456 2 BEGIN
: 366      0457 2 $WAKE ();
: 367      0458 2 RETURN;
: 368      0459 1 END;                               ! of TAKE_AST
```

```
                                0000 00000 TAKE_AST:
                                .WORD  Save nothing           : 0424
                                CLRQ   -(SP)                   : 0457
00000000G 00                   7E 7C 00002                  CALLS  #2, SYSSWAKE
                                02 FB 00004                    RET                               : 0459
                                04 0000B
```

; Routine Size: 12 bytes, Routine Base: _BASSCODE + 01B0

```
370 0460 1 ROUTINE SLEEP_HANDLER (
371 0461 1     SIG,
372 0462 1     MECH,
373 0463 1     ENBL
374 0464 1 ) =
375 0465 1
376 0466 1 ++
377 0467 1 FUNCTIONAL DESCRIPTION:
378 0468 1
379 0469 1     Handle an UNWIND while in BASSSLEEP. This is needed so that the
380 0470 1     ASTs will not fire after their storage has been removed from the
381 0471 1     stack.
382 0472 1
383 0473 1 FORMAL PARAMETERS:
384 0474 1
385 0475 1     SIG.rl.a      Address of the signal vector. This contains
386 0476 1                the condition.
387 0477 1     MECH.rl.a    Address of the mechanism vector. This contains
388 0478 1                the status of the frame that signalled.
389 0479 1     ENBL.rl.a    Address of the enable vector. This contains
390 0480 1                some the local variables ASSIGNED_CHAN, and TIMER_REQID.
391 0481 1
392 0482 1 IMPLICIT INPUTS:
393 0483 1
394 0484 1     NONE
395 0485 1
396 0486 1 IMPLICIT OUTPUTS:
397 0487 1
398 0488 1     NONE
399 0489 1
400 0490 1 COMPLETION CODES:
401 0491 1
402 0492 1     Always SSS_RESIGNAL, but this is ingored when we are
403 0493 1     unwinding.
404 0494 1
405 0495 1 SIDE EFFECTS:
406 0496 1
407 0497 1     Arranges that the ASTs will not fire after this routine
408 0498 1     has completed its execution.
409 0499 1
410 0500 1 --
411 0501 1
412 0502 2 BEGIN
413 0503 2
414 0504 2 MAP
415 0505 2     SIG : REF VECTOR,
416 0506 2     MECH : REF VECTOR,
417 0507 2     ENBL : REF VECTOR;
418 0508 2
419 0509 2 BIND
420 0510 2     ASSIGNED_CHAN = .ENBL [1],
421 0511 2     TIMER_REQID = .ENBL [2],
422 0512 2     EVENT_FLAG = .ENBL [3];
423 0513 2
424 0514 2 ++
425 0515 2     If this is the UNWIND condition, cancel the SETIMR and QIO.
426 0516 2 --
```

```
! Handler for BASSSLEEP
! signal args
! mechanism args
! variables passed from BASSSLEEP
```

```
! signal vector
! mechanism vector
! enable vector
```

```
427 0517 2
428 0518 2
429 0519 2
430 0520 2
431 0521 2
432 0522 2
433 0523 2
434 0524 2
435 0525 3
436 0526 4
437 0527 4
438 0528 4
439 0529 4
440 0530 4
441 0531 4
442 0532 4
443 0533 4
444 P 0534 4
445 0535 4
446 0536 4
447 0537 4
448 0538 4
449 0539 4
450 0540 4
451 0541 4
452 0542 4
453 0543 5
454 0544 5
455 0545 5
456 0546 5
457 0547 5
458 0548 5
459 0549 4
460 0550 4
461 0551 4
462 0552 4
463 0553 5
464 0554 5
465 0555 5
466 0556 5
467 0557 5
468 0558 5
469 0559 4
470 0560 4
471 0561 3
472 0562 3
473 0563 3
474 0564 3
475 0565 3
476 0566 3
477 0567 3
478 0568 4
479 0569 4
480 0570 4
481 0571 4
482 0572 4
483 0573 4

IF (LIB$MATCH_COND (SIG [1], %REF (SS$_UNWIND)))
THEN
  BEGIN
    + Turn off the QIO and SETIMR. We need to do this while no ASTs can go off
    because we are modifying ASSIGNED_CHAN, and TIMER_REQID.
    $SETAST (ENBFLG = 0);
    BEGIN
      LOCAL
        DASSGN_STATUS,
        CANTIM_STATUS,
        EF_STATUS,
        QIO_STATUS;
      QIO_STATUS = $QIO (EFN = .EVENT_FLAG, CHAN = .ASSIGNED_CHAN, FUNC = (IO$_SETMODE OR IO$_OUTBAND OR
        P1 = 0);
      IF ( NOT .QIO_STATUS) THEN LIB$STOP (.QIO_STATUS);
      EF_STATUS = LIB$FREE_EF (EVENT_FLAG);
      IF .ASSIGNED_CHAN NEQ 0
      THEN
        BEGIN
          DASSGN_STATUS = $DASSGN (CHAN = .ASSIGNED_CHAN);
          IF ( NOT .DASSGN_STATUS) THEN LIB$STOP (.DASSGN_STATUS);
          ASSIGNED_CHAN = 0;
          END;
        IF .TIMER_REQID NEQ 0
        THEN
          BEGIN
            CANTIM_STATUS = $CANTIM (REQIDT = TIMER_REQID);
            IF ( NOT .CANTIM_STATUS) THEN LIB$STOP (.CANTIM_STATUS);
            TIMER_REQID = 0;
            END;
          END;
          $SETAST (ENBFLG = 1);
    + Make sure there are not any $WAKE hanging around. They could have appeared
    as a result of one of the ASTs timer or QIO going off just before we turned
    it off.
    BEGIN
      LOCAL
        WAKE_STATUS,
        HIBER_STATUS;
```

```

: 484      0574 4      WAKE_STATUS = $WAKE ();
: 485      0575 4
: 486      0576 4      IF ( NOT .WAKE_STATUS) THEN LIB$STOP (.WAKE_STATUS);
: 487      0577 4
: 488      0578 4      HIBER_STATUS = $HIBER;
: 489      0579 4
: 490      0580 4      IF ( NOT .HIBER_STATUS) THEN LIB$STOP (.HIBER_STATUS);
: 491      0581 4
: 492      0582 3      END;
: 493      0583 2      END;
: 494      0584 2
: 495      0585 2      RETURN (SS$_RESIGNAL);
: 496      0586 1      END;

```

! of HANDLER

```

                                .EXTRN  SYS$SETAST
                                001C 00000 SLEEP_HANDLER:
                                .WORD    Save R2,R3,R4
                                MOVAB    SYS$SETAST, R4
                                MOVAB    LIB$STOP, R3
                                MOVL     ENBL, R2
                                MOVZWL   #2336, -(SP)
                                PUSHL    SP
                                ADDL3    #4, SIG, -(SP)
                                CALLS    #2, LIB$MATCH_COND
                                BLBS     R0, 1$
                                BRW      8$
                                CLRL     -(SP)
                                CALLS    #1, SYS$SETAST
                                CLRQ     -(SP)
                                CLRQ     -(SP)
                                CLRQ     -(SP)
                                CLRQ     -(SP)
                                CLRL     -(SP)
                                MOVZWL   #3107, -(SP)
                                PUSHL    @4(R2)
                                PUSHL    @12(R2)
                                CALLS    #12, SYS$QIO
                                BLBS     QIO_STATUS, 2$
                                PUSHL    QIO_STATUS
                                CALLS    #1, LIB$STOP
                                PUSHL    12(R2)
                                CALLS    #1, LIB$FREE_EF
                                TSTL     @4(R2)
                                BEQL     4$
                                PUSHL    @4(R2)
                                CALLS    #1, SYS$DASSGN
                                BLBS     DASSGN_STATUS, 3$
                                PUSHL    DASSGN_STATUS
                                CALLS    #1, LIB$STOP
                                CLRL     @4(R2)
                                TSTL     @8(R2)
                                BEQL     6$
                                CLRL     -(SP)
                                PUSHL    8(R2)
                                0460
                                0510
                                0518
                                0525
                                0535
                                0537
                                0539
                                0541
                                0544
                                0546
                                0548
                                0551
                                0554

```

BASSSLEEP
3-003

I 16
16-Sep-1984 01:14:56
14-Sep-1984 11:56:40

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASSSLEEP.B32;1

Page 14
(5)

00000000G	00	02	FB	00084	CALLS	#2, SYSSCANTIM	:	
	05	50	E8	0008B	BLBS	CANTIM_STATUS, 5\$:	0556
		50	DD	0008E	PUSHL	CANTIM_STATUS	:	
	63	01	FB	00090	CALLS	#1, LIB\$STOP	:	
		08	B2	D4 00093	CLRL	@8(R2)	:	0558
		01	DD	00096	PUSHL	#1	:	0562
	64	01	FB	00098	CALLS	#1, SYSS\$SETAST	:	
		7E	7C	0009B	CLRL	-(SP)	:	0574
00000000G	00	02	FB	0009D	CALLS	#2, SYSS\$WAKE	:	
	05	50	E8	000A4	BLBS	WAKE_STATUS, 7\$:	0576
		50	DD	000A7	PUSHL	WAKE_STATUS	:	
	63	01	FB	000A9	CALLS	#1, LIB\$STOP	:	
00000000G	00	00	FB	000AC	CALLS	#0, SYSS\$HIBER	:	0578
	05	50	E8	000B3	BLBS	HIBER_STATUS, 8\$:	C580
		50	DD	000B6	PUSHL	HIBER_STATUS	:	
	63	01	FB	000B8	CALLS	#1, LIB\$STOP	:	
	50	0918	8F	3C 000BB	MOVZWL	#2328, R0	:	0585
			04	000C0	RET		:	0586

; Routine Size: 193 bytes, Routine Base: _BAS\$CODE + 01BC

: 497 0587 1 END
: 498 0588 1
: 499 0589 0 ELUDOM

! end of module BASSSLEEP

PSECT SUMMARY

Name	Bytes	Attributes
_BAS\$CODE	637	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	25	0	581	00:01.1

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD, INITIAL, OPTIMIZE)/NOTRACE/LIS=LIS\$:BASSSLEEP/OBJ=OBJ\$:BASSSLEEP MSRC\$:BASSSLEEP/UPDATE=(ENH\$:BASSSLEEP)

BASSSLEEP
3-003

J 16
16-Sep-1984 01:14:56

VAX-11 Bliss-32 V4.0-742

Page 15

: Size: 617 code + 20 data bytes
: Run Time: 00:12.7
: Elapsed Time: 00:29.9
: Lines/CPU Min: 2784
: Lexemes/CPU-Min: 21976
: Memory Used: 143 pages
: Compilation Complete

0031 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

